程序说明：

1：编程语言为 Fortran 90

2：本程序可以求解任意空间杆系结构

3：节点位移编码为（X, Y, Z, θx, θy, θz），荷载编码为（Fx, Fy, Fz, Mx, My, Mz）

4：单元输入信息为：起点号，终点号，EA, EIy, EIz, GIx, α(α为参考坐标系与整体坐标系夹角

5：程序为使编程简化，采用了多文件技术

程序清单：

程序由以下四个文件组成

1：Lxz_Tools.f90 ----- 主要为一些工具函数

2：TypeDef.f90 ----- 变量定义，单元属性分析

3：SolveDisp.f90 ----- 矩阵求解

4：3dframe.f90 ----- 整体控制模块

1：Lxz_Tools.f90

```
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
```

2：TypeDef.f90

```
include 'Lxz_Tools.f90'
module TypeDef
    use Lxz_Tools
    implicit none
    integer(ikind),parameter :: NDOF=6,NNode=2

    type::typ_Joint
     real(rkind) :: X,Y,Z
     integer(ikind):: GDOF(NDOF)
    end type typ_Joint

    type::typ_Element
     integer(ikind):: JointNo(NNode)
     real(rkind):: EIy,EIz,EA,GIp,Length,CosA,CosB,CosC,A
     real(rkind)::EK(NDOF*NNode,NDOF*NNode),ET(NDOF*NNode,NDOF*NNode)
     integer(ikind)::GlbDOF(NDOF*NNode)
     real(rkind)::EForce(NDOF*NNode),ELForce(NDOF*NNode)
    end type typ_Element

    type::typ_JointLoad
     integer(ikind)::JointNo,LodDOF
     real(rkind)::LodVal
    end type typ_JointLoad

    type::typ_ElemLoad
     integer (ikind):: ElemNo,Indx
     real(rkind)::Pos,LodVal
    end type typ_ElemLoad
```

```
contains

subroutine SetElemProp(Elem,Joint)
 type(typ_Element),intent(inout)::Elem(:)
 type(typ_Joint),intent(in)::Joint(:)
 integer(ikind):: i,EJ1,EJ2
 real(rkind)::T(3,3),x,y,z
 real(rkind)::cx,cy,cz,cs,ca,sa
 do i=1,size(Elem)
     Elem(i)%EK=zero;Elem(i)%ET=zero;T=zero
     EJ1=Elem(i)%JointNo(1);EJ2=Elem(i)%JointNo(2)
     Elem(i)%GlbDOF(1:6)=Joint(EJ1)%GDOF(:)
     Elem(i)%GlbDOF(7:12)=Joint(EJ2)%GDOF(:)
     x=Joint(EJ2)%X-Joint(EJ1)%X
     y=Joint(EJ2)%Y-Joint(EJ1)%Y
     z=Joint(EJ2)%Z-Joint(EJ1)%Z
     Elem(i)%Length=sqrt(x**2+y**2+z**2)
     Elem(i)%CosA=x/Elem(i)%Length
     Elem(i)%CosB=y/Elem(i)%Length
     Elem(i)%CosC=z/Elem(i)%Length
     cx=Elem(i)%CosA;cy=Elem(i)%CosB;cz=Elem(i)%CosC
     ca=cos(Elem(i)%A);sa=sin(Elem(i)%A)
     cs=sqrt(cx**2+cy**2)
     if(cs.NE.zero) then
         T(1,1)=cx;T(1,2)=cy;T(1,3)=cz;
         T(2,1)=-(ca*cy+sa*cx*cz)/cs;
         T(2,2)=(ca*cx-sa*cy*cz)/cs;
         T(2,3)=cs*sa;
         T(3,1)=(sa*cy-ca*cx*cz)/cs;
         T(3,2)=-(sa*cx+ca*cy*cz)/cs;
         T(3,3)=cs*ca;
     else
         T(1,3)=one;
         T(2,1)=-sa;T(2,2)=ca;
         T(3,1)=-ca;T(3,2)=-sa;
     end if
     Elem(i)%ET(1:3,1:3)=T;Elem(i)%ET(4:6,4:6)=T;
     Elem(i)%ET(7:9,7:9)=T;Elem(i)%ET(10:12,10:12)=T;
     T=zero
     T(1,1)=Elem(i)%EA/Elem(i)%Length
     T(2,2)=12D0*Elem(i)%EIz/(Elem(i)%Length**3)
     T(3,3)=12D0*Elem(i)%EIy/(Elem(i)%Length**3)
     Elem(i)%EK(1:3,1:3)=T
     Elem(i)%EK(7:9,7:9)=T
```

```
        Elem(i)%EK(1:3,7:9)=-T
        Elem(i)%EK(7:9,1:3)=transpose(-T)
        T=zero
        T(1,1)=Elem(i)%GIp/Elem(i)%Length
        T(2,2)=4D0*Elem(i)%EIy/Elem(i)%Length
        T(3,3)=4D0*Elem(i)%EIz/Elem(i)%Length
        Elem(i)%EK(4:6,4:6)=T;Elem(i)%EK(10:12,10:12)=T
        T=zero
        T(2,3)=6D0*Elem(i)%EIz/(Elem(i)%Length**2)
        T(3,2)=-6D0*Elem(i)%EIy/(Elem(i)%Length**2)
        Elem(i)%EK(1:3,4:6)=T;Elem(i)%EK(1:3,10:12)=T
        Elem(i)%EK(4:6,1:3)=transpose(T)
        Elem(i)%EK(10:12,1:3)=transpose(T)
        Elem(i)%EK(7:9,10:12)=-T
        Elem(i)%EK(10:12,7:9)=-transpose(T)
        T=zero
        T(2,3)=6D0*Elem(i)%EIy/(Elem(i)%Length**2)
        T(3,2)=-6D0*Elem(i)%EIz/(Elem(i)%Length**2)
        Elem(i)%EK(4:6,7:9)=T
        Elem(i)%EK(7:9,4:6)=transpose(T)
        T=zero
        T(1,1)=-Elem(i)%GIp/Elem(i)%Length
        T(2,2)=2D0*Elem(i)%EIy/Elem(i)%Length
        T(3,3)=2D0*Elem(i)%EIz/Elem(i)%Length
        Elem(i)%EK(4:6,10:12)=T
        Elem(i)%EK(10:12,4:6)=transpose(T)
        !call matprint(Elem(i)%EK,10)
        !write(16,*) Elem(i)%ET
        !write(*,*)
    end do
    end subroutine SetElemProp
end module TypeDef

3：SolveDisp.f90
include 'TypeDef.f90'
module Solve
    use TypeDef
    type :: typ_Kcol
        real(rkind),pointer :: row(:)
    end type typ_Kcol
    contains
    !================================================
    subroutine SolveDisp (GDisp, Elem,Joint,GLoad)
    !================================================
```

```
real(rkind),intent(out)         :: GDisp(:)
type (typ_Element),intent(in)   :: Elem(:)
type (typ_Joint),intent(in)     :: Joint(:)
real(rkind)                    :: GLoad(:)  !?
integer(ikind) NElem,NGlbDOF
type (typ_Kcol),allocatable     ::Kcol(:)
NElem = size(Elem,dim=1)
NGlbDOF = size(GDisp,dim=1)
allocate (Kcol(NGlbDOF))


call SetMatBand()
call GStifMat()
call BandSolv()
contains

  !-------------------------
subroutine SetMatBand()
  !-------------------------
    integer (ikind) :: minDOF
    integer (ikind),allocatable  :: Row1(:)
    integer (ikind) :: ie,j
    integer (ikind),allocatable::ELocVec(:)
    allocate (Row1(NGlbDOF))
    allocate(ELocVec(size(Elem(1)%GlbDOF)))
    Row1=NGlbDOF
    do ie=1,NElem
             ELocVec(:)=Elem(ie)%GlbDOF(:)
             minDOF=minval(ELocVec,mask=ELocVec>0)
             where(ELocVec>0)
                 Row1(ELocVec)=min(Row1(ELocVec),minDOF)
             end where
         end do
         do j=1,NGlbDOF
             allocate (Kcol(j)%row(Row1(j):j))
             Kcol(j)%row=Zero
         end do
     return
end subroutine SetMatBand

  !-------------------------------------
subroutine BandSolv()
  !-------------------------------------
    integer(ikind)::row1,ncol,row,j,ie
```

```
    real(rkind)::diag(1:NGlbDOF),s
    !integer (ikind)::ELocVec(NNode*NDOF)
    ncol=NGlbDOF
    diag(1:ncol)=(/(Kcol(j)%row(j),j=1,ncol)/)
    do j=2,ncol
        row1=lbound(Kcol(j)%row,1)
        do ie=row1,j-1
            row=max(row1,lbound(Kcol(ie)%row,1))

s=sum(diag(row:ie-1)*Kcol(ie)%row(row:ie-1)*Kcol(j)%row(row:ie-1))
            Kcol(j)%row(ie)=(Kcol(j)%row(ie)-s)/diag(ie)
        end do
        s=sum(diag(row1:j-1)*Kcol(j)%row(row1:j-1)**2)
        diag(j)=diag(j)-s
    end do
    do ie=2,ncol
        row1=lbound(Kcol(ie)%row,dim=1)
        GLoad(ie)=GLoad(ie)-sum(Kcol(ie)%row(row1:ie-1)*GLoad(row1:ie-1))
    end do
    GLoad(:)=GLoad(:)/diag(:)
    do j=ncol,2,-1
        row1=lbound(Kcol(j)%row,dim=1)
        GLoad(row1:j-1)=GLoad(row1:j-1)-GLoad(j)*Kcol(j)%row(row1:j-1)
    end do
    GDisp(:)=GLoad(:)
    return
end subroutine BandSolv

  !---------------------
subroutine GStifMat()
  !---------------------
    integer(ikind)::ie,j,JGDOF
    integer (ikind),allocatable::ELocVec(:)
    real(rkind),allocatable::EK(:,:),ET(:,:)
    integer(ikind)::n
    n=size(Elem(1)%GlbDOF)
    allocate(ELocVec(n))
    allocate(EK(n,n))
    allocate(ET(n,n))
    do IE=1,NElem
        EK=Elem(IE)%EK
        ET=Elem(IE)%ET
        EK = matmul(transpose(ET),matmul(EK,ET))
        !write(16,*) EK
```

```
            !write(16,*)
            ELocVec(:)=Elem(IE)%GlbDOF(:)
            do j=1,12
                JGDOF=ELocVec(j)
                if (JGDOF==0) cycle
                where (ELocVec>0.and.ELocVec<=JGDOF)
                    Kcol(JGDOF)%row(ELocVec)=Kcol(JGDOF)%row(ELocVec)+EK(:,j)
                end where
            end do
        end do
        return
      end subroutine GStifMat

  end subroutine SolveDisp
end module Solve
```

4：3dframe.f90

```
include 'SolveDisp.f90'

module DispMethod
    use Solve
    implicit none

    contains
    subroutine GLoadVec(Elem,Joint,JLoad,ELoad,GLoad)
     type (typ_Element),intent(inout)::Elem(:)
     type (typ_Joint),intent(in)::Joint(:)
     type (typ_JointLoad),intent(in)::JLoad(:)
     type (typ_ElemLoad),intent(in)::ELoad(:)
     real(rkind)::GLoad(:),EF(NNode*NDOF)
     integer (ikind)::i,m,n
     real(rkind)::a,b,l,q
     do i=1,size(JLoad)
          n=JLoad(i)%JointNo
          m=JLoad(i)%LodDOF
          m=Joint(n)%GDOF(m)
          GLoad(m)=GLoad(m)+JLoad(i)%LodVal
     end do
     do i=1,size(ELoad)
          n=ELoad(i)%ElemNo
          l=Elem(n)%Length
          a=ELoad(i)%Pos
          q=ELoad(i)%LodVal
          b=l-a
```

```
        EF=zero
        if(ELoad(i)%Indx.eq.1) then
            EF(5)=q*a*a*(6D0-8D0*a/l+3D0*a*a/(l*l))/12D0
            EF(11)=-q*(a**3D0)*(4D0-3D0*a/l)/(12D0*l)
            EF(3)=-0.5D0*q*a*(2D0-2D0*a*a/(l*l)+a**3D0/l**3D0)
            EF(9)=-0.5D0*q*a**3D0*(2D0-a/l)/l**2D0
        end if
        if(ELoad(i)%Indx.eq.2) then
            EF(5)=q*a*b**2D0/l**2D0
            EF(11)=-q*a*a*b/l**2D0
            EF(3)=-q*b**2D0*(1D0+2D0*a/l)/l**2D0
            EF(9)=-q*a**2D0*(1D0+2D0*b/l)/l**2D0
        end if
        if(ELoad(i)%Indx.eq.-1) then
            EF(6)=-q*a*a*(6D0-8D0*a/l+3D0*a*a/(l*l))/12D0
            EF(12)=q*(a**3D0)*(4D0-3D0*a/l)/(12D0*l)
            EF(2)=-0.5D0*q*a*(2D0-2D0*a*a/(l*l)+a**3D0/l**3D0)
            EF(8)=-0.5D0*q*a**3D0*(2D0-a/l)/l**2D0
        end if
        if(ELoad(i)%Indx.eq.-2) then
            EF(6)=-q*a*b**2D0/l**2D0
            EF(12)=q*a*a*b/l**2D0
            EF(2)=-q*b**2D0*(1D0+2D0*a/l)/l**2D0
            EF(8)=-q*a**2D0*(1D0+2D0*b/l)/l**2D0
        end if
        if(ELoad(i)%Indx.eq.3) then
            EF(5)=q*b*(2D0-3D0*b/l)/l
            EF(11)=q*a*(2D0-3D0*a/l)/l
            EF(3)=-6D0*q*a*b/l**3D0
            EF(9)=6D0*q*a*b/l**3D0
        end if
        if(ELoad(i)%Indx.eq.-3) then
            EF(6)=q*b*(2D0-3D0*b/l)/l
            EF(12)=q*a*(2D0-3D0*a/l)/l
            EF(2)=6D0*q*a*b/l**3D0
            EF(8)=-6D0*q*a*b/l**3D0
        end if

        Elem(n)%ELForce=Elem(n)%ELForce+EF
        EF=matmul(transpose(Elem(n)%ET),EF)
        where(Elem(n)%GlbDOF>0)
            GLoad(Elem(n)%GlbDOF)=GLoad(Elem(n)%GlbDOF)-EF(:)
        end where
    end do
```

```
      i=1
    end subroutine GLoadVec

end module DispMethod

program DFrame
    use DispMethod
    implicit none
    integer(ikind):: NElem,NJoint,NGlbDOF,NJLoad,NELoad
    type(typ_Element),allocatable::Elem(:)
    type(typ_Joint),allocatable::Joint(:)
    type(typ_JointLoad),allocatable::JLoad(:)
    type(typ_ElemLoad),allocatable::ELoad(:)
    real(rkind),allocatable::Disp(:),GLoad(:)
    call Input_Data()
    call SetElemProp(Elem,Joint)
    call GLoadVec(Elem,Joint,JLoad,ELoad,GLoad)
    write(16,*) "整体荷载向量"
    write(16,*) GLoad
    write(16,*) "整体节点位移"

    call SolveDisp(Disp,Elem,Joint,GLoad)
    write(16,*) Disp
    write(16,*)
    call Output_Results()

    stop

    contains
    subroutine Input_Data()
     integer(ikind)::i
     open(5,file='data.ipt',status='old',position='rewind')
     open(16,file='data.opt',position='rewind')
     read(5,*) NElem,NJoint,NGlbDOF,NJLoad,NELoad
     allocate (Elem(NElem))
     allocate (Joint(NJoint))
     allocate (JLoad(NJLoad))
     allocate (ELoad(NELoad))
     allocate (Disp(NGlbDOF))
     allocate (GLoad(NGlbDOF))
     read(5,*) (Joint(i),i=1,NJoint)
     read(5,*) (Elem(i)%JointNo,Elem(i)%EA,Elem(i)%EIy,&
                    Elem(i)%EIz,Elem(i)%GIp,Elem(i)%A,i=1,NElem)
    if(NJLoad>0) read(5,*) (JLoad(i),i=1,NJLoad)
```

```
 if(NELoad>0) read(5,*) (ELoad(i),i=1,NELoad)
end subroutine Input_Data

subroutine Output_Results()
 integer(ikind)::i
 real(rkind) :: EDisp(12)
 do i=1,size(Elem)
      EDisp=zero
      where(Elem(i)%GlbDOF>0)
           EDisp(:)=Disp(Elem(i)%GlbDOF)
      end where
      write(16,*) "单元   ",itoc(i),"位移"
      write(16,*) EDisp
      EDisp=matmul(Elem(i)%ET,EDisp)
      Elem(i)%EForce=matmul(Elem(i)%EK,EDisp)+Elem(i)%ELForce
      write(16,*) "单元   ",itoc(i),"内力"
      write(16,*) Elem(i)%EForce

 end do

end subroutine Output_Results
end program DFrame
```

程序考题：

1：自选考题

如图所示空间刚架，各杆材料，几何参数相同已知 EA=5.4E6kN，GIp=6.08E4kNm^2，EIy=1.62E5kNm^2，EIz=3.75E4 kNm^2 。

输入文件：

```
3,4,6,2,6
0,0,5,1,2,3,4,5,6
0,5,5,0,0,0,0,0,0
5,0,5,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0
1, 2, 5.4e6, 1.62e5, 3.75e4, 6.08e4,0
1, 3, 5.4e6, 1.62e5, 3.75e4, 6.08e4,0
4, 1, 5.4e6, 1.62e5, 3.75e4, 6.08e4,0
1,4,15
1,5,20
1, 1,     5, -14
1,-2, 2.5, -16
2, 2, 2.5, -17
2,-3, 2.5, 9
3, 2, 2.5, -12
3,-2, 2.5, 12
```

输出文件：

整体荷载向量

14.0000000000000 3.30000000000000 -43.5000000000000
-6.66666666666667 23.1250000000000 -12.2500000000000

整体节点位移

1.426524186386400E-005 4.688151946271983E-006 -3.518505521174383E-005
-3.109492948038260E-005 8.222154207338749E-005 -1.685671589626063E-004

单元 1 位移

1.426524186386400E-005 4.688151946271983E-006 -3.518505521174383E-005
-3.109492948038260E-005 8.222154207338749E-005 -1.685671589626063E-004
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000

单元 1 内力

5.06320410197374 6.43154069862663 33.2438311631497
0.999813951612392 -23.7687688593765 4.81459805434703
-5.06320410197374 9.56845930137337 36.7561688368503
-0.999813951612392 32.5496130436281 -12.6568945612139

单元 2 位移

1.426524186386400E-005 4.688151946271983E-006 -3.518505521174383E-005
-3.109492948038260E-005 8.222154207338749E-005 -1.685671589626063E-004
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000

单元 2 内力

15.4064612129731 1.19977291634312 4.75602846553365
-0.378114342481452 1.39890679934362 -2.76482140136174
-15.4064612129731 -1.19977291634312 12.2439715344663
0.378114342481452 17.3209508729881 -0.236314016922647

单元 3 位移

0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
1.426524186386400E-005 4.688151946271983E-006 -3.518505521174383E-005
-3.109492948038260E-005 8.222154207338749E-005 -1.685671589626063E-004

单元 3 内力

37.9998596286833 -5.73702298168314 3.02507948565351
2.04977665298529 -2.72667667731152 -7.07576942531071
-37.9998596286833 -6.26297701831686 8.97492051434649
-2.04977665298529 17.6012792490440 8.39065451689503

2：规定考题：

题目见附后。

输入文件：

5,6,12,6,0
0,0,2,1,2,3,4,5,6
2.5,0,2,7,8,9,10,11,12
0,1.5,2,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0
4,1.5,0,0,0,0,0,0,0
4,-1.5,0,0,0,0,0,0,0
1, 2, 3.39e2, 5.47e3, 5.47e3, 4.46e3,0
1, 3, 3.39e2, 5.47e3, 5.47e3, 4.46e3,0
1, 4, 3.39e2, 5.47e3, 5.47e3, 4.46e3,0
2, 5, 3.39e2, 5.47e3, 5.47e3, 4.46e3,0
2, 6, 3.39e2, 5.47e3, 5.47e3, 4.46e3,0
1,4,7.9e3
1,5,8.1e3
1,6,8.3e3
2,1,-7.1e3
2,2,-7.3e3
2,3,-7.5e3

输出文件：

整体荷载向量

0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
7900.00000000000 8100.00000000000 8300.00000000000
-7100.00000000000 -7300.00000000000 -7500.00000000000

0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
整体节点位移
-0.309630428262080 0.263278832839125 -0.760790620509566
0.822951752899139 0.771280204272966 -0.134994574555115
-2.26398531672088 -1.37476947766886 -1.34992089009757
0.738654146991511 -0.856298991704140 -0.312286073228411

单元 1 位移
-0.309630428262080 0.263278832839125 -0.760790620509566
0.822951752899139 0.771280204272966 -0.134994574555115
-2.26398531672088 -1.37476947766886 -1.34992089009757
0.738654146991511 -0.856298991704140 -0.312286073228411
单元 1 内力
265.010522875014 4532.61529287078 2921.36335388699
150.386928939208 -90.5609115608313 6053.68291518564
-265.010522875014 -4532.61529287078 -2921.36335388699
-150.386928939208 -7212.84747315664 5277.85531699130
单元 2 位移
-0.309630428262080 0.263278832839125 -0.760790620509566
0.822951752899139 0.771280204272966 -0.134994574555115
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
单元 2 内力
59.5010162216422 4052.84693504433 -2792.40934371062
2293.27314070495 -906.723717789231 2547.35498607226
-59.5010162216422 -4052.84693504433 2792.40934371062
-2293.27314070495 5095.33773335516 3531.91541649423
单元 3 位移
-0.309630428262080 0.263278832839125 -0.760790620509566
0.822951752899139 0.771280204272966 -0.134994574555115
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
单元 3 内力
-128.954010176371 -4592.11630909242 -3787.83641216932
-301.037901257905 5897.28777085588 -6842.88935327156
128.954010176371 4592.11630909242 3787.83641216932
301.037901257905 1678.38505348276 -2341.34326491327
单元 4 位移
-2.26398531672088 -1.37476947766886 -1.34992089009757
0.738654146991511 -0.856298991704140 -0.312286073228411
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
单元 4 内力
-110.007111701298 567.768288251598 -2922.16720868377

235.123266125875 2143.77783413962 294.278508063015
110.007111701298 -567.768288251598 2922.16720868377
-235.123266125875 6375.73037712511 1361.03628004385

单元 5 位移

-2.26398531672088 -1.37476947766886 -1.34992089009757
0.738654146991511 -0.856298991704140 -0.312286073228411
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000

单元 5 内力

54.4800128621408 -4705.27180303564 -3422.91841323718
1583.03862012604 4833.64228117416 -5833.81735887245
-54.4800128621408 4705.27180303564 3422.91841323718
-1583.03862012604 5145.79402260895 -7884.28940896397

内力图：