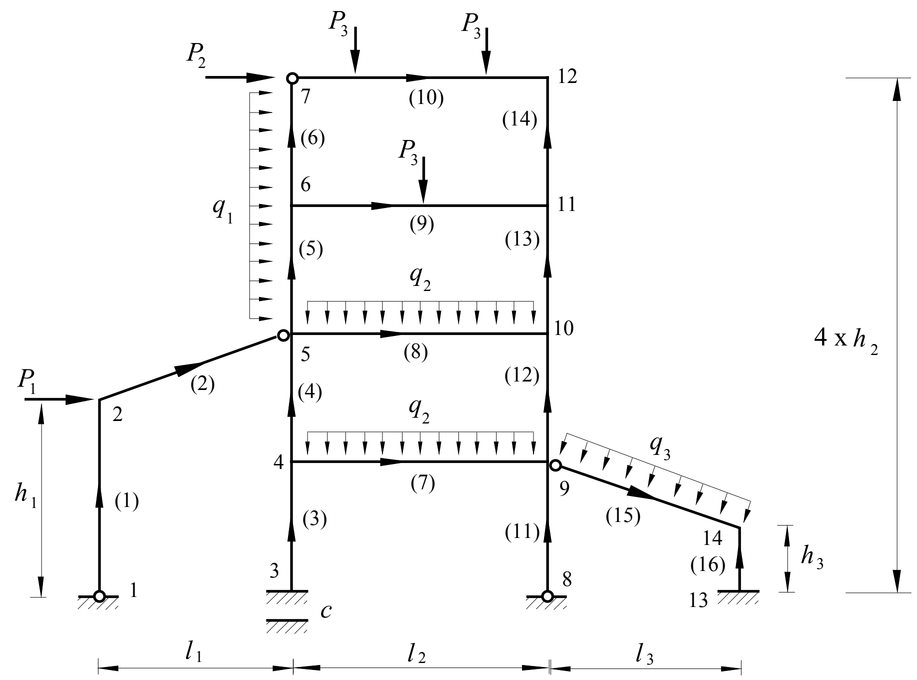
程序结构力学编程大作业题目

班级 结六(2) 姓名 陆新征 学号 960133



指定的数据

i	跨长 l_i (m)	层高 h_i (m)	集中力 P_i (kN)	均布荷载 q_i (kN / m)
1	5	6	8	4
2	5	4	15	9
3	4	3	28	3

其它：

柱刚度： $EA = 10^5 (\text{kN})$， $EI = 1.5 \times 10^4 (\text{kN} \cdot \text{m}^2)$

梁刚度： $EA = 10^6 (\text{kN})$， $EI = 1.0 \times 10^4 (\text{kN} \cdot \text{m}^2)$

支座沉降： $c = 0.01(\text{m})$

源程序清单

```
!     Last change:  123   9 Dec 99   10:02 pm
!  Rules:
!  * indent = 3
!  * all key words in lower case
!  * variable names in both uper and lower cases
!  * modules are highlighted by ****, Subroutines by =====,
!    internal sub in a sub by -----
!  * arguments in Subs : 'out' first, then a space, and 'in' follows
!  * first character: N-Number of; G-Global;
!***************
module NumKind
!***************
   ! This module defines the kind of integer and real numbers.
   ! Every module, subroutine or func must use this module.
   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
end module NumKind

!**************
module TypeDef
!**************
   use NumKind
   implicit none
   integer (ikind),parameter :: NDOF=3,NNode=2

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

   type :: typ_Element
      integer (ikind)   :: JointNo(NNode)
      real (rkind)      :: EI,EA,Length,CosA,SinA
      integer(ikind)    :: GlbDOF(NDOF*NNode)
   end type typ_Element

   type :: typ_Kcol//变带宽存储刚度矩阵
```

```
   real(rkind),pointer :: row(:)
end type typ_Kcol

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)//设置单元属性
!====================================
   type (typ_Element),intent(in out) :: Elem(:)
   type (typ_Joint),intent(in) :: Joint(:)
      integer(ikind) :: i,N
      real(rkind)  ::  x1,x2,y1,y2
      N=size(Elem,dim=1)
      do i=1,N
          x1=Joint(Elem(i)%JointNo(1))%X
          y1=Joint(Elem(i)%JointNo(1))%Y
          x2=Joint(Elem(i)%JointNo(2))%X
          y2=Joint(Elem(i)%JointNo(2))%Y
          Elem(i)%Length=sqrt((x2-x1)**2+(y2-y1)**2)
          Elem(i)%CosA=(x2-x1)/Elem(i)%Length
          Elem(i)%SinA=(y2-y1)/Elem(i)%Length
          Elem(i)%GlbDOF(1:3)=Joint(Elem(i)%JointNo(1))%GDOF
          Elem(i)%GlbDOF(4:6)=Joint(Elem(i)%JointNo(2))%GDOF
      end do
      return
   end subroutine SetElemProp
!==========================================
subroutine TransMatrix (ET, CosA,SinA)//设置坐标转换矩阵
!==========================================
   real(rkind),intent(out) :: ET(:,:)
   real(rkind),intent(in)  :: CosA,SinA
   ! ET could be 2x2, 3x3 or 6x6 depending on size(ET)
   ET = Zero
   ET(1,1) = CosA
```

```
      ET(1,2) = SinA
      ET(2,1) =-SinA
      ET(2,2) = CosA
      if (size(ET,dim=1) > 2) ET(3,3) = One
      if (size(ET,dim=1) > 3) ET(4:6,4:6) = ET(1:3,1:3)
      return
   end subroutine TransMatrix
end module TypeDef

!****************
module DispMethod
!****************
   use TypeDef
   implicit none
   real (rkind),allocatable :: lxz_EForce1(:,:)//单元内力

   contains
   subroutine SolveDisp (Disp, Elem,Joint,JLoad,ELoad)
      real(rkind),intent(out)        :: Disp(:)
      type (typ_Element),intent(in)  :: Elem(:)
      type (typ_Joint),intent(in)    :: Joint(:)
      type (typ_JointLoad),intent(in out) :: JLoad(:)
      type (typ_ElemLoad),intent(in out)  :: ELoad(:)
      real(rkind),allocatable        :: GLoad(:)  !?
      integer(ikind) ::NElem,NGlbDOF
      type (typ_Kcol),allocatable    ::Kcol(:)
      NElem = size(Elem,dim=1)
      NGlbDOF = size(Disp,dim=1)
      allocate (Kcol(NGlbDOF))
      allocate (lxz_EForce1(NElem+5,NNode*NDOF))
      lxz_EForce1=zero
      allocate (GLoad(NGlbDOF))
      GLoad=zero

      call SetMatBand()//得到刚度矩阵带宽
      call GLoadVec()//得到整体荷载向量
      call GStifMat()//得到整体刚度矩阵
      call BandSolv()//求解位移
      return

      contains

      subroutine SetMatBand()//得到刚度矩阵带宽
        integer (ikind) :: minDOF
```

```
    integer (ikind),allocatable  :: Row1(:)
    integer (ikind) :: ie,j
    integer (ikind)::ELocVec(NNode*NDOF)
    allocate (Row1(NGlbDOF))
    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()//得到整体荷载向量
  !----------------------------------------
     integer(ikind)::row1,ncol,row,j,ie
     real(rkind)::diag(1:NGlbDOF),s
     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
         Disp(:)=GLoad(:)
         return
      end subroutine BandSolv

      !---------------------
      subroutine GStifMat()
      !---------------------
         integer(ikind)::ie,j,JGDOF
         real(rkind)::ET(NNode*NDOF,NNode*NDOF)
         real(rkind)::EK(NNode*NDOF,NNode*NDOF)
         integer (ikind)::ELocVec(NNode*NDOF)
         do IE=1,NElem
            call EStifMat(EK, Elem(IE)%Length,Elem(IE)%EI,Elem(IE)%EA)
            call TransMatrix(ET, Elem(IE)%CosA,Elem(IE)%SinA)
            EK = matmul(transpose(ET),matmul(EK,ET))
            ELocVec(:)=Elem(IE)%GlbDOF(:)
            do j=1,6
               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

      subroutine GLoadVec ()
          if(size(JLoad)>0) call ProJointLoad(GLoad,JLoad,Joint)
          if(size(ELoad)>0) call ProElemLoad(GLoad,ELoad,Elem,Joint)
          return
      end subroutine GLoadVec

   end subroutine SolveDisp

   !====================================
   subroutine EStifMat (EK, ELen,EI,EA)
        real(rkind),intent (out) :: EK(:,:)
        real(rkind),intent (in) :: ELen,EI,EA
        real(rkind) :: a1,a2,a3,a4
        a1=EA/ELen
        a2=12.0D0*EI/(ELen**3)
```

```
      a3=6.0D0*EI/(ELen**2)
      a4=4.0D0*EI/ELen
      EK=zero
      EK(1,1)=a1
      EK(4,1)=-a1
      EK(1,4)=-a1
      EK(4,4)=a1
      EK(2,2)=a2
      EK(5,5)=a2
      EK(5,2)=-a2
      EK(2,5)=-a2
      EK(2,3)=a3
      EK(3,2)=a3
      EK(6,2)=a3
      EK(2,6)=a3
      EK(3,5)=-a3
      EK(5,3)=-a3
      EK(5,6)=-a3
      EK(6,5)=-a3
      EK(3,3)=a4
      EK(6,6)=a4
      EK(3,6)=a4/2.0D0
      EK(6,3)=a4/2.0D0
      return
end subroutine EStifMat
!===================================

!=========================================
subroutine ElemDisp (EDisp, IE,Disp,Elem)
      real(rkind), intent(out) ::EDisp(:)
      integer(ikind),intent(in) ::IE
      real(rkind), intent(in):: Disp(:)
      type(typ_Element),intent(in out):: Elem(:)
      integer (ikind):: i
      do i=1,6
          if (Elem(IE)%GlbDOF(i).eq.0)  then
                EDisp(i)=0.0D0
          else
                EDisp(i)=Disp(Elem(IE)%GlbDOF(i))
          end if
      end do
      return
end subroutine ElemDisp
```

```
subroutine ElemForce (EForce, IE,Disp,Elem,ELoad)
    real(rkind), intent(out) ::EForce(:)
    integer(ikind),intent(in) ::IE
    real(rkind), intent(in):: Disp(:)
    type(typ_Element),intent(in out):: Elem(:)
    type(typ_ElemLoad),intent(in out):: ELoad(:)
    real(rkind) ::lxz_EDisp(NNode*NDOF)
    real(rkind)::ET(NNode*NDOF,NNode*NDOF),EK(NNode*NDOF,NNode*NDOF)
    integer (ikind) :: i
    call TransMatrix(ET,Elem(IE)%CosA,Elem(IE)%SinA)
    do i=1,6
        if (Elem(IE)%GlbDOF(i).eq.0) then
             lxz_EDisp(i)=0
        else
              lxz_EDisp(i)=Disp(Elem(IE)%GlbDOF(i))
        end if
    end do
    lxz_EDisp=matmul(ET,lxz_EDisp)
    call EStifMat(EK,Elem(IE)%Length,Elem(IE)%EI,Elem(IE)%EA)
    EForce=matmul(EK,lxz_EDisp)
    EForce=lxz_EForce1(IE,:)+EForce
    EForce(1:3)=-EForce(1:3)
    return
end subroutine ElemForce

subroutine ProJointLoad(GLoad,JLoad,Joint)
   type (typ_JointLoad),intent (in) :: JLoad(:)
   real (rkind),intent (in out) :: GLoad(:)
   type (typ_Joint),intent (in) :: Joint(:)
   integer (ikind):: i,n
   n=size(JLoad)
   do i=1,n
     GLoad(Joint(JLoad(i)%JointNo)%GDOF(JLoad(i)%LodDOF))=&
       GLoad(Joint(JLoad(i)%JointNo)%GDOF(JLoad(i)%LodDOF))+&
       JLoad(i)%LodVal
   end do
   return
end subroutine ProJointLoad

subroutine ProElemLoad(GLoad,ELoad,Elem,Joint)
    type (typ_ElemLoad),intent (in) :: ELoad(:)
    real (rkind),intent (in out) :: GLoad(:)
    type (typ_Element),intent(in) :: Elem(:)
    type (typ_Joint),intent (in) :: Joint(:)
```

```
        integer (ikind):: i,n,ie
        real (rkind) :: l,a,q,ET(NDOF*NNode,NDOF*NNode)
        real (rkind) :: lxz_EForce2(NNode*NDOF)
        INTEGER(ikind) :: EVec(6)
        n=size(ELoad)
        do i=1,n
             ie=ELoad(i)%ElemNo
             l=Elem(ie)%Length
             a=ELoad(i)%Pos
             a=a*l
             q=ELoad(i)%LodVal
             call TransMatrix (ET, Elem(ie)%CosA,Elem(ie)%SinA)
             if(ELoad(i)%Indx.eq.1) then
                 lxz_EForce2(1)=0.0D0
                 lxz_EForce2(4)=0.0D0

lxz_EForce2(2)=-0.5D0*q*a*(2.0D0-2.0D0*(a**2)/(l**2)+(a**3)/(l**3))
                 lxz_EForce2(5)=-0.5D0*q*(a**3)*(2D0-a/l)/(l**2)
                 lxz_EForce2(3)=-q*(a**2)*(6D0-8D0*a/l+3D0*(a**2)/(l**2))/12D0
                 lxz_EForce2(6)=q*(a**3)*(4D0-3D0*a/l)/(12D0*l)
             end if
             if(ELoad(i)%Indx.eq.2) then
                   lxz_EForce2(1)=zero
                   lxz_EForce2(4)=zero
                   lxz_EForce2(2)=-q*((l-a)**2)*(l+2*a/l)/l**2
                   lxz_EForce2(5)=-q*(a**2)*(l+2*(l-a)/l)/l**2
                   lxz_EForce2(3)=-q*a*((l-a)**2)/l**2
                   lxz_EForce2(6)=q*a*a*(l-a)/l**2
               end if
             if(ELoad(i)%Indx.eq.3) then
                   if(a<l/2) then
                        lxz_EForce2(1)=Elem(ie)%EA*q/l
                        lxz_EForce2(4)=-lxz_EForce2(1)
                   else
                        lxz_EForce2(1)=-Elem(ie)%EA*q/l
                        lxz_EForce2(4)=-lxz_EForce2(1)
                   end if
                   lxz_EForce2(2)=zero
                   lxz_EForce2(5)=zero
                   lxz_EForce2(3)=zero
                   lxz_EForce2(6)=zero
               end if

             if(ELoad(i)%Indx.eq.4) then
```

```
        lxz_EForce2(1)=zero
        lxz_EForce2(4)=zero
        if(a.lt.l/2) then
              lxz_EForce2(2)=12D0*Elem(ie)%EI*q/l**3
              lxz_EForce2(5)=-lxz_EForce2(2)
              lxz_EForce2(3)=6D0*Elem(ie)%EI*q/l**2
              lxz_EForce2(6)=6D0*Elem(ie)%EI*q/l**2
        else
              lxz_EForce2(2)=-12D0*Elem(ie)%EI*q/l**3
              lxz_EForce2(5)=-lxz_EForce2(2)
              lxz_EForce2(3)=-6D0*Elem(ie)%EI*q/l**2
              lxz_EForce2(6)=-6D0*Elem(ie)%EI*q/l**2
        end if
    end if
 if(ELoad(i)%Indx.eq.5) then
       lxz_EForce2(1)=zero
       lxz_EForce2(4)=zero
       lxz_EForce2(2)=6*q*a*(l-a)/l**2
       lxz_EForce2(5)=-6*q*a*(l-a)/l**2
       lxz_EForce2(3)=q*(l-a)*(2-3*(l-a)/l)/l
       lxz_EForce2(6)=q*a*(2-3*a/l)/l
 end if
 if(ELoad(i)%Indx.eq.6) then
       lxz_EForce2(1)=zero
       lxz_EForce2(4)=zero
       lxz_EForce2(2)=-q*a*(2-3*(a**2)/(l**2)+1.6*(a**3)/(l**3))/4
       lxz_EForce2(5)=-q*a*a*(3-1.6*a/l)/(4*l*l)
       lxz_EForce2(3)=-q*a*a*(2-3*a/l+1.2*a*a/(l**2))/6
       lxz_EForce2(6)=q*a*a*(1-0.8*a/l)/(4*l)
  end if
 if(ELoad(i)%Indx.eq.7) then
       lxz_EForce2(1)=-q*(l-a)/l
       lxz_EForce2(4)=-q*a/l
       lxz_EForce2(2)=zero
       lxz_EForce2(5)=zero
       lxz_EForce2(3)=zero
       lxz_EForce2(6)=zero
 end if
 if(ELoad(i)%Indx.eq.8) then
       lxz_EForce2(1)=-q*a*(1-0.5*a/l)
       lxz_EForce2(4)=-0.5*q*a*a/l
       lxz_EForce2(2)=zero
       lxz_EForce2(5)=zero
       lxz_EForce2(3)=zero
```

```
                lxz_EForce2(6)=zero
            end if
            lxz_EForce1(ie,:)=lxz_EForce1(ie,:)+lxz_EForce2
            lxz_EForce2=matmul(transpose(ET),lxz_EForce2)
            EVec(1:3)=Joint(Elem(ie)%JointNo(1))%GDOF
            EVec(4:6)=Joint(Elem(ie)%JointNo(2))%GDOF
            where(EVec>0)
                GLoad(EVec)=GLoad(EVec)-lxz_EForce2(:)
            end where
        end do
        return
    end subroutine ProElemLoad
end module DispMethod

program SM_90            ! main prog
    use DispMethod       ! displacement method module
    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(:)
    call Input_Data ()                 ! internal sub, see below
    call SetElemProp (Elem, Joint)
    call SolveDisp (Disp, Elem,Joint,JLoad,ELoad)
    call Output_Results ()             ! internal sub, see below
    stop
    contains

    subroutine Input_Data ()
        integer (ikind) :: i,ie
        open (5,file='SM90.IPT',status='OLD',position='REWIND')
        read(5,*) NElem
        read(5,*) NElem,NJoint,NGlbDOF,NJLoad,NELoad
        allocate (Joint(NJoint))
        allocate (Elem(NElem))
        allocate (JLoad(NJLoad))
        allocate (ELoad(NELoad))
        allocate (Disp(NGlbDOF))
        Disp=zero
        read(5,*) (Joint(i),i=1,NJoint)
        read(5,*) (Elem(ie)%JointNo,Elem(ie)%EA,Elem(ie)%EI,ie=1,NElem)
        if(NJLoad>0) read(5,*) (JLoad(i),i=1,NJLoad)
```

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

   subroutine Output_Results ()
      real (rkind):: EDisp(NDOF*NNode),EForce(NDOF*NNode)
      integer (ikind) :: i
      open (55,file='SMCAI90.OUT',position='REWIND')
      write(55,*) 10,0
      do i=1,size(Elem)
         call ElemDisp(EDisp,i,Disp,Elem)
         write(55,*) EDisp(1),EDisp(2),EDisp(3),EDisp(4),EDisp(5),EDisp(6)
      end do
      do i=1,size(Elem)
         call ElemForce(EForce,i,Disp,Elem,ELoad)
         write(55,*)
EForce(1),EForce(2),-1D0*EForce(3),EForce(4),EForce(5),-1D0*EForce(6)
      end do
      return
  end subroutine Output_Results
end program SM_90
```

求解器教学版输入文件

```
N,1,0,0
N,2,0,6
N,3,5,0
N,4,5,4
N,5,5,8
N,6,5,12
N,7,5,16
N,8,10,0
N,9,10,4
N,10,10,8
N,11,10,12
N,12,10,16
N,13,14,0
N,14,14,3
E,1,2,1,1,0,1,1,1
E,2,5,1,1,1,1,1,0
E,3,4,1,1,1,1,1,1
E,4,5,1,1,1,1,1,1
E,5,6,1,1,1,1,1,1
```

```
E,6,7,1,1,1,1,1,0
E,4,9,1,1,1,1,1,1
E,5,10,1,1,1,1,1,1
E,6,11,1,1,1,1,1,1
E,7,12,1,1,0,1,1,1
E,8,9,1,1,0,1,1,1
E,9,10,1,1,1,1,1,1
E,10,11,1,1,1,1,1,1
E,11,12,1,1,1,1,1,1
E,9,14,1,1,0,1,1,1
E,13,14,1,1,1,1,1,1
NSUPT,1,3,0,0,0
NSUPT,3,6,0,0,-0.01,0
NSUPT,8,3,0,0,0
NSUPT,13,6,0,0,0,0
ECHAR,1,1,1E5,1.5E4,0,0,-1
ECHAR,2,2,1E6,1.E4,0,0,-1
ECHAR,3,6,1E5,1.5E4,0,0,-1
ECHAR,7,10,1E6,1.E4,0,0,-1
ECHAR,11,14,1E5,1.5E4,0,0,-1
ECHAR,15,15,1E6,1.0E4,0,0,-1
ECHAR,16,16,1E5,1.5E4,0,0,-1
NLOAD,2,1,8,0
ELOAD,7,3,9,0,1,90
NLOAD,7,1,15,0
ELOAD,8,3,9,0,1,90
ELOAD,15,3,3,0,1,90
ELOAD,5,3,4,0,1,90
ELOAD,6,3,4,0,1,90
ELOAD,9,1,28,0.5,90
ELOAD,10,1,28,0.25,90
ELOAD,10,1,28,0.75,90
```

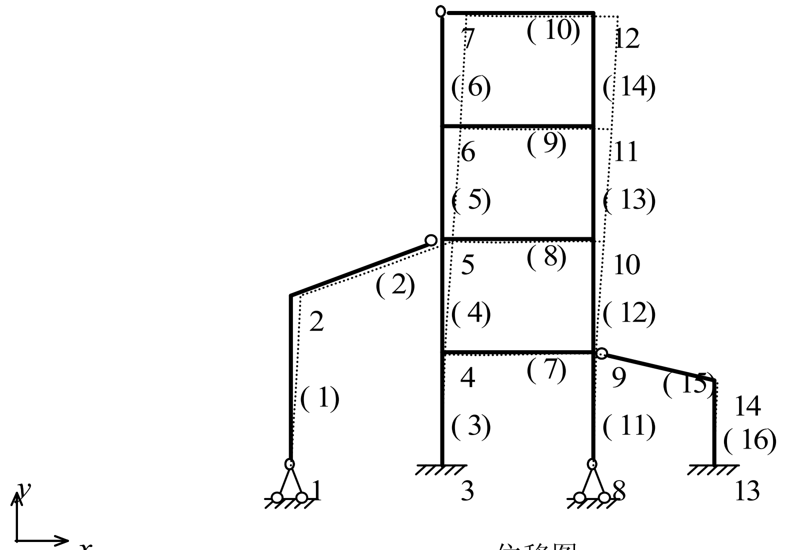
输出文件:

smcai90.out

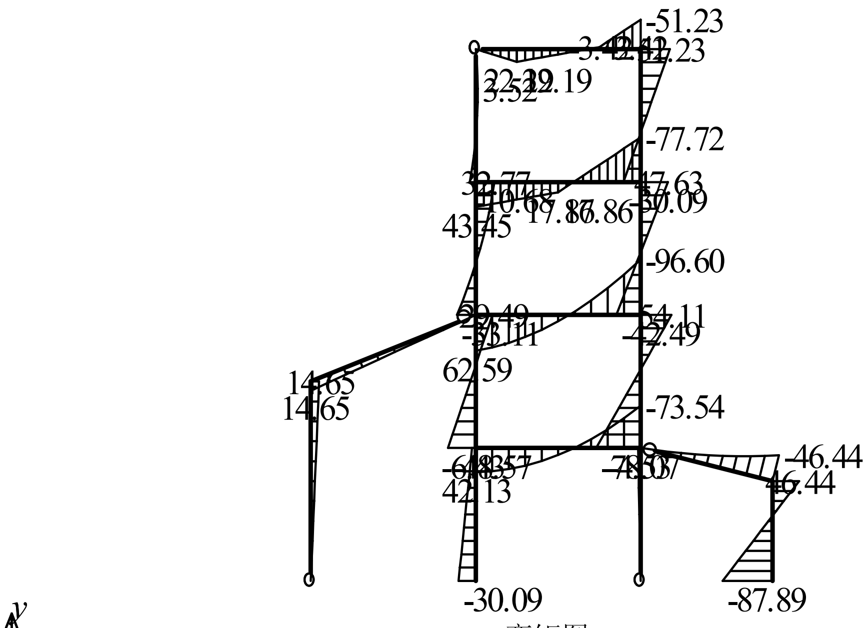
```
              10              0
   0.000000000000000       0.000000000000000      -0.754423897989734D-002
   0.394045534886535D-001  0.424352156292456D-004 -0.461379878453205D-002
   0.394045534886535D-001  0.424352156292456D-004 -0.461379878453205D-002
   0.433473610464348D-001 -0.988560141570041D-002 -0.668572932158675D-003
   0.000000000000000       0.000000000000000       0.000000000000000
   0.118424006145884D-001 -0.993011153700349D-002 -0.486975934158929D-002
```

0.118424006145884D-001	-0.993011153700349D-002	-0.486975934158929D-002
0.433473610464348D-001	-0.988560141570041D-002	-0.741365352488932D-002
0.433473610464348D-001	-0.988560141570041D-002	-0.741365352488932D-002
0.761021700660480D-001	-0.101863378103218D-001	-0.603569153074340D-002
0.761021700660480D-001	-0.101863378103218D-001	-0.603569153074340D-002
0.101197708923611	-0.108965065491829D-001	-0.603742575065874D-002
0.118424006145884D-001	-0.993011153700349D-002	-0.486975934158929D-002
0.117744048725457D-001	-0.637104162250815D-002	-0.334597041858593D-002
0.433473610464348D-001	-0.988560141570041D-002	-0.741365352488932D-002
0.432947870609699D-001	-0.116038418875641D-001	-0.653964953549428D-002
0.761021700660480D-001	-0.101863378103218D-001	-0.603569153074340D-002
0.760911721465516D-001	-0.146631054929427D-001	-0.585326912813617D-002
0.101197708923611	-0.108965065491829D-001	-0.335271040411678D-002
0.101096058515256	-0.161929367540815D-001	-0.303493731470559D-002
0.000000000000000	0.000000000000000	-0.274241661791167D-002
0.117744048725457D-001	-0.637104162250815D-002	-0.334597041858593D-002
0.117744048725457D-001	-0.637104162250815D-002	-0.334597041858593D-002
0.432947870609699D-001	-0.116038418875641D-001	-0.653964953549428D-002
0.432947870609699D-001	-0.116038418875641D-001	-0.653964953549428D-002
0.760911721465516D-001	-0.146631054929427D-001	-0.585326912813617D-002
0.760911721465516D-001	-0.146631054929427D-001	-0.585326912813617D-002
0.101096058515256	-0.161929367540815D-001	-0.303493731470559D-002
0.117744048725457D-001	-0.637104162250815D-002	0.367630892694431D-002
0.129336374760023D-001	-0.875352738180895D-003	-0.414489753919728D-002
0.000000000000000	0.000000000000000	0.000000000000000
0.129336374760023D-001	-0.875352738180895D-003	-0.414489753919728D-002
0.707253593820759	-2.44203349613774	0.256045185054177D-014
0.707253593820759	-2.44203349613774	-14.6522009768265
-4.89777495699647	2.72084541542992	-14.6522009768264
-4.89777495699647	2.72084541542992	0.697662413950972D-014
1.74721157491288	-5.91435543209004	30.0903083951399
1.74721157491288	-5.91435543209004	6.43288666677977
1.11275303257685	-19.5135038406258	48.5666108686267
1.11275303257685	-19.5135038406258	-29.4874044938765
-7.51840986553402	-24.4703344297281	33.1066447147423
-7.51840986553402	-8.47033442972809	-32.7746930041700
-17.7542184715290	-10.6699183290079	10.6796733160317
-17.7542184715290	5.33008167099211	0.125233157177718D-012
-13.5991484085356	0.634458542336049	-42.1337242018469
-13.5991484085356	45.6344585423361	73.5385685098334
-10.5147970929665	9.33841649193153	-62.5940492086189
-10.5147970929665	54.3384164919315	96.5980332510388
-2.19958389928183	10.2358086059949	-43.4543663202017
-2.19958389928183	38.2358086059949	77.7246767097728

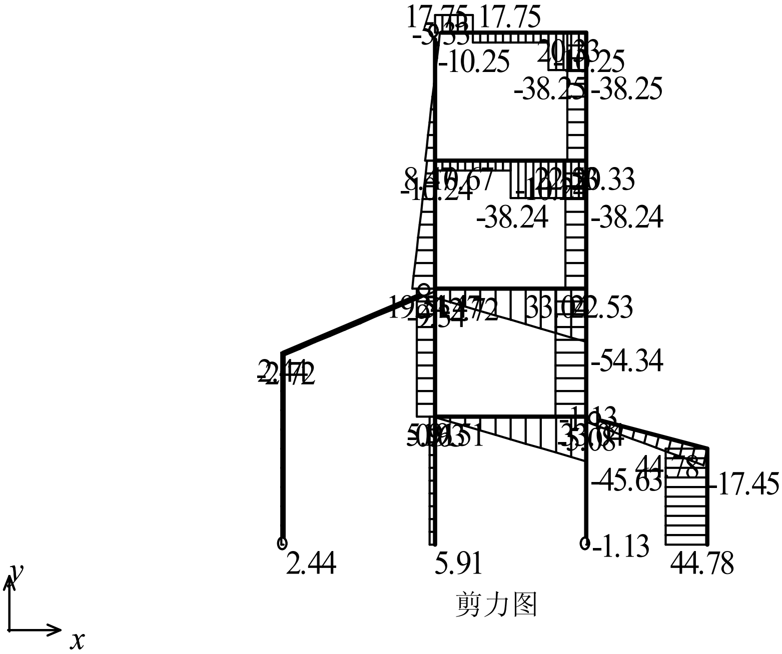
-20.3300816709914	-17.7542184715290	-0.355271367880050D-014
-20.3300816709914	38.2457815284711	51.2289076423552
-159.276040562704	1.13166337626423	0.888872309090516D-014
-159.276040562704	1.13166337626423	4.52665350505692
-130.820006626398	-33.0444626632420	78.0652220148903
-130.820006626398	-33.0444626632420	-54.1126286380777
-76.4815901344660	-22.5296655702770	42.4854046129611
-76.4815901344660	-22.5296655702770	-47.6332576681469
-38.2457815284710	-20.3300816709953	30.0914190416259
-38.2457815284710	-20.3300816709953	-51.2289076423553
-50.5152041471156	5.07831374631363	-0.799360577730113D-014
-50.5152041471156	17.4476306231666	46.4384239760772
-29.1784246060298	-44.7752744480424	87.8873993680499
-29.1784246060298	-44.7752744480424	-46.4384239760772



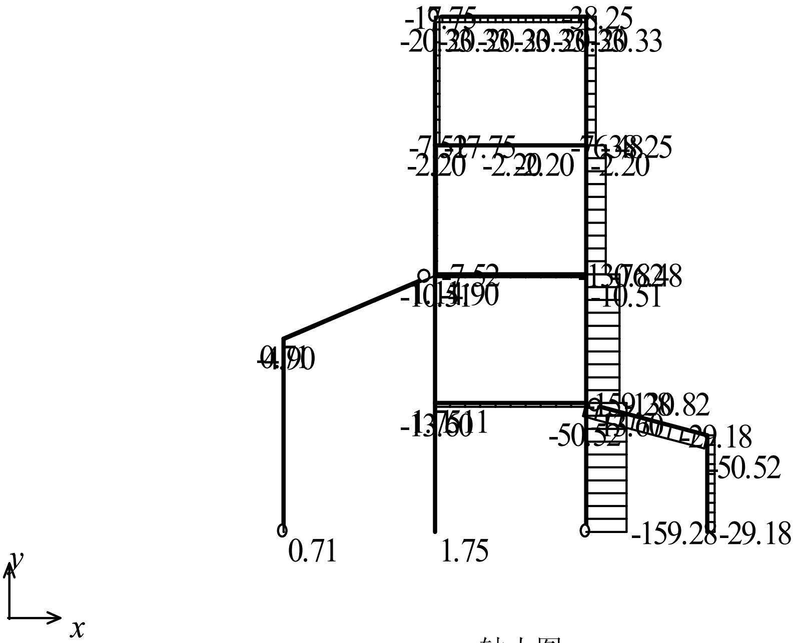
位移图



弯矩图



剪力图



轴力图

注:以上计算结果与求解器结果一致